SEKOLAH TINGGI
MANAJEMEN INFORMATIKA &
TEKNIK KOMPUTER
SURABAYA

STIKOM
SURABAYA

PROSIDING
SNASTI 2009
SEMINAR NASIONAL
SISTEM & TEKNOLOGI INFORMASI

2 DESEMBER 2009

SNASTI
Seminar Nasional
Sistem & Teknologi Informasi

ISBN: 978-979-8968-303

SEKOLAH TINGGI
MANAJEMEN INFORMATIKA &
TEKNIK KOMPUTER
SURABAYA

PROSIDING

SNASTI 2009

SEMINAR NASIONAL
SISTEM & TEKNOLOGI INFORMASI

urabaya, 2 Desember 2009
ampus STIKOM SURABAYA
l. Raya Kedung Baruk 98
urabaya

:ditor:
holiq
Gede Arya Utama
Vinarti
chmad Yanu Aliffianto
M Arifin
'uwono Marta Dinata
Susijanto TR
)ian Arisanti
Kurniawan Jatmika

Call For Papers
SNASTI
Seminar Nasional
Sistem & Teknologi Informasi

Diterbitkan oleh:
Bagian Penelitian Akademik
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER SURABAYA

SNASTI 2009
Susunan Panitia

Keynote Speaker
1. Prof. Dr. Richardus Eko Indrajit
   (Ketua APTIKOM Pusat)
2. Prof. Dr. Abdullah Shahab
   (Dosen ITS)

Reviewer/Komite Program
- Prof. Achmad Benny Mutiara (Universitas Gunadarma)
- Ir. Kridanto Surendro, M.Sc., Ph.D. (ITB)
- Dr. Ir. Joko Lianto Buliali, M.Sc. (ITS)
- Dr. Iping Supriana Suwardi (ITB)
- Dr. Jusak (STIKOM SURABAYA)
- Karsam, MA., Ph.D. (STIKOM SURABAYA)
- Prof. Dr. Ir. Mauridhi Heri P., M.Eng. (ITS)
- Dr. Daniel Siahaan (ITS)

Pelindung
Dr. Y. Jangkung Karyantoro, MBA

Ketua Pelaksana
Achmad Yanu Aliffianto

Komite Pelaksana
- Sholiq, S.T., M.Kom.
- Ir. I Gede Arya Utama., M.MT.
- Achmad Yanu Aliffianto, S.T, M.B.A.
- Tutut Wurijanto, M.Kom.
- Titik Lusiani, M.Kom.
- Anjik Sukmaaji, S.Kom., M.Eng.

Alamat Sekretariat:
Bagian Penelitian Akademik STIKOM SURABAYA
Jalan Raya Kedung Baruk 98, Surabaya 60298
Telp: 031.8721731, Faksimili: 031.8710218
Email: snasti@stikom.edu, snastistikom@yahoo.co.id
Website: http://snasti.stikom.edu

KATA PENGANTAR

Seminar Nasional Sistem dan Teknologi Informasi 2009 (SNASTI 2009) merupakan temu ilmiah nasional tahunan yang diselenggarakan oleh STIKOM (STMIK) Surabaya, di mana tahun ini adalah tahun ke-4 sejak diadakan mulai tahun SNASTI 2006. Konferensi ini kami maksudkan sebagai sarana desiminasi hasil-hasil penelitian atau kajian kritis terhadap Sistem dan Teknologi Informasi dengan skala nasional, sekaligus sebagai sarana komunikasi antar peneliti, praktisi, dan akademisi Teknologi Informasi.

Tahun ini, SNASTI 2009 mengambil tema: **IT for Life** yang merupakan sebuah kerinduan sekaligus keresahan kita agar kemajuan Teknologi Informasi dapat dimanfaatkan sebesar-besarnya untuk kehidupan bersama yang lebih baik.

Suksesnya acara SNASTI 2009 tidak lepas dari peran serta dan kerja sama yang baik dari berbagai pihak, untuk itu perkenankan kami mengucapkan terima kasih yang setulus-tulusnya kepada:

1. Prof Dr. Ir. Eko Indrajit M.Sc, M.BA dan Dr. Ir. Abdullah Sahab, M.Sc atas partisipasinya sebagai *keynote speaker*.
2. Komite Program: Prof. Dr. Ir. Mauridi Heri P, M.Eng (ITS), Prof Achmad Benny Mutiara (Univ Gunadarma), Ir. Kridanto Surendro, M.Sc, Ph.D (ITB), Dr. Ir. Joko Lianto Buliali M.Sc (ITS), Dr. Ir. Iping Supriana Suwandi (ITB), Dr. Daniel Siahaan (ITS), Dr. Jusak (STIKOM), dan Karsam, MA, Ph.D (STIKOM)
3. Para pemakalah yang mempercayakan artikelnya dimuat dan dipresentasikan di acara SNASTI 2009.
4. Para sponsor yang berpartisipasi.
5. Pimpinan, dosen, karyawan, dan mahasiswa STIKOM Surabaya.
6. Panitia SNASTI 2009
7. Semua pihak yang tidak dapat kami sebutkan satu per satu.

Semoga acara ini bermanfaat bagi kemajuan dan perkembangan sistem dan teknologi informasi Indonesia. Akhirnya, kami mohon maaf yang sebesar-besarnya atas kesalahan-kesalahan dalam penyajian buku prosiding ini atau pada penyelenggaraan acara SNASTI 2009.

Surabaya, 2 Desember 2009
Redaksi SNASTI 2009

DAFTAR ISI

APLIKASI *ANT COLONY SYSTEM* (ACS) PADA *TRAVELLING SALESMAN PROBLEM*

Rina Refianti[1)], Pipit Dewi Arnesia[2)]
[1, 2] Fakultas Ilmu Komputer & Teknologi Informasi, Jurusan Sistem Informasi
Universitas Gunadarma
Margonda Raya 100, Pondok Cina, Depok
{rina, pdarnesia}@staff.gunadarma.ac.id

Abstraksi: *Ant Colony System (ACS)* telah diterapkan dalam berbagai bidang, salah satunya adalah untuk mencari solusi optimal pada *Travelling Salesman Problem (TSP)*. Dengan memberikan sejumlah n kota, TSP dapat didefinisikan sebagai suatu permasalahan dalam menemukan jalur terpendek dengan mengunjungi setiap kota yang ada hanya sekali. Penelitian ini dilakukan dengan mengimplementasikan ACS ke dalam bentuk kode-kode program berbahasa Java. Kemudian dilakukan percobaan untuk membandingkan antara ACS dengan metodologi lainnya yang juga mengimplementasikan *agent* di dalamnya. Dari hasil percobaan diketahui bahwa secara garis besar ACS terbukti merupakan metodologi yang paling optimal dalam menemukan jalur terpendek. Penelitian ini telah berhasil membuktikan keoptimalan ACS dalam menemukan solusi terhadap TSP.

Kata kunci: *Genetic Algorithm, Ant System, Software Agent, Pheromone,State Transition Rule*

Salah satu paradigma dalam rekayasa perangkat lunak (*software engineering*) adalah *software agent* [Romi]. Sesuai dengan namanya, *software* ini menggunakan *agent* yang mempunyai kemampuan untuk melakukan tugas tertentu yang telah didelegasikan kepadanya. Di dalam kamus Webster's New World Dictionary [Guralnik], *agent* didefinisikan sebagai: *A person or thing that acts or is capable of acting or is empowered to act, for another*. Berdasarkan definisi tersebut, Caglayan [Caglayan et.al.] mendefinisikan *sofware agent* sebagai suatu entitas *software* komputer yang memungkinkan pengguna (*user*) untuk mendelegasikan tugas kepadanya secara mandiri (*autonomously*).

Dalam perkembangan aplikasi dan penelitian tentang *agent*, bagaimanapun juga dalam suatu komunitas sebuah sistem tidak dapat dihindari akan dibutuhkannya lebih dari satu *agent*, seiring dengan semakin kompleksnya tugas yang dikerjakan oleh sistem tersebut. Paradigma pengembangan sistem di mana dalam suatu komunitas terdapat beberapa *agent* yang saling berinteraksi, bernegosiasi, dan berkoordinasi satu sama lain dalam menjalankan pekerjaan disebut dengan *Multi Agent System* (MAS) [Romi]. Dan salah satu sistem yang menerapkan paradigma MAS adalah *Ant Colony System* (ACS). ACS, yang dikemukakan oleh Marco Dorigo dan Luca M. Gambardella pada tahun 1997, merupakan sistem yang dihasilkan melalui pengembangan terhadap *ant system* dengan tujuan untuk meningkatkan performanya jika diterapkan pada masalah yang lebih kompleks. *Ant System* sendiri merupakan suatu metodologi yang dikemukakan pada tahun 1991 oleh Marco Dorigo, yang juga dikenal dengan *Ant Colony Optimization* (ACO). ACO merupakan suatu algoritma yang mengambil inspirasi dari riset atas perilaku semut riil yang di dalamnya terdapat sekumpulan semut buatan, dinamai *ants*, yang bekerja sama untuk mencari solusi terhadap suatu masalah optimisasi.

Semut adalah seranggga sosial yang hidupnya berkoloni. Perilaku semut ditentukan oleh keselamatan dari keseluruhan koloni, semut secara individu tidaklah begitu berguna [Walkowiak]. Koloni semut telah diketahui mampu untuk menemukan jalur terpendek dari sarang mereka menuju ke sumber makanan dan kembali lagi. Hal ini telah diamati bahwa pada saat semut berjalan, ia meninggalkan sejumlah informasi, disebut *pheromone*, di tempat yang dilaluinya dan menandai jalur tersebut. Dengan perantara *pheromone* inilah terjadi komunikasi tidak langsung dan juga pertukaran informasi antar semut selagi membangun suatu solusi. Bentuk komunikasi tidak langsung yang diperlihatkan oleh semut ini disebut *stigmergy*.

Ant system telah diterapkan di banyak permasalahan optimisasi kombinatorial, sebagai contoh *traveling salesman problem* (TSP), *quadratic assignment problem, job scheduling, vehicle routing, graph coloring, network routing* [Dorigo, Di Caro, dan Gambardella]. ACS yang dikembangkan berdasar pada *ant system* terdahulu ini diarahkan pada peningkatan efisiensinya ketika diterapkan pada TSP yang lebih rumit di mana *ant system* memiliki kelemahan jika diimplementasikan pada TSP dengan jumlah kota yang lebih banyak. "Walaupun *ant system* bermanfaat dalam menemukan solusi yang

optimal bagi TSP dengan jumlah kota yang sedikit (sampai dengan 30 kota), waktu yang dibutuhkan untuk mencapai hasil tersebut membuatnya tidak mungkin lagi untuk diterapkan pada masalah yang lebih besar [Dorigo dan Gambardella]".

Berdasarkan hal tersebut di atas, penulis berkeinginan untuk menganalisa dan mengimplementasikan ACS, sebagai perbaikan dari *ant system*, pada TSP dengan menunjukkan karakteristik dari ACS yang membedakannya dengan *ant system* serta memperlihatkan cara kerjanya dalam menemukan solusi yang optimal pada TSP.

Dengan menetapkan sejumlah *n* kota, TSP dapat didefinisikan sebagai permasalahan dalam mencari jalur terpendek dengan melakukan tur tertutup (yang dimulai dari suatu kota dan kembali ke kota tersebut) dimana setiap kota yang ada hanya dikunjungi sekali. TSP direpresentasikan dengan menggunakan graf dimana kota-kota yang ada diwakili dengan simpul-simpul dan jalur-jalur yang dilewati merupakan ruas-ruas yang menghubungkan simpul-simpul yang ada pada graf tersebut.

Dalam paper ini, penelitian yang dilakukan dibatasi hanya pada permasalahan TSP simetris, artinya jarak simpul A ke simpul B adalah sama dengan jarak simpul B ke simpul A dan graf yang direpresentasikan sebagai permasalahannya merupakan graf yang terhubung secara penuh, artinya pada setiap simpul yang ada pasti terdapat ruas yang menghubungkannya dengan simpul-simpul lainnya yang terdapat pada graf tersebut.

Paper ini dibuat dengan tujuan untuk memperkenalkan ACS sebagai metodologi yang dapat digunakan untuk mencari solusi optimal pada TSP dan untuk membuktikan keoptimalan dari ACS dibandingkan dengan metodologi lain yang juga diuji cobakan untuk dijadikan sebagai perbandingan. Pencarian solusi optimal yang dimaksud di sini adalah mendapatkan jalur terpendek dari suatu graf dalam waktu yang seminimal mungkin.

Penelitian dilakukan dengan mengumpulkan data-data dan membaca artikel-artikel yang berhubungan dengan MAS, *ant system*, dan ACS serta bahasa pemrograman yang digunakan.

Selain itu, penelitian juga dilakukan dengan mengadakan percobaan untuk membandingkan antara ACS dengan metodologi yang lain, yaitu *ant system* {asal mula dari ACS} dan *genetic algorithms* {metodologi yang juga menerapkan konsep *agent*}.

Paper ini diorganisasikan sebagai berikut pada seksi 2 akan diuraikan teori-teori tentang *agent* dan MAS, *ant system*, *dan* ACS. Dilanjutkan dengan seksi hasil dan diskusi. Pada Seksi ini dijelaskan secara rinci mengenai ACS yang diterapkan pada TSP dan pengimplementasiannya ke dalam bentuk kode-kode program berbahasa Java. Dalam seksi ini juga ditampilkan hasil perbandingan antara ACS dengan metodologi yang lain.

KAJIAN PUSTAKA

Agent dan Multi Agent System

Agent dan Karakteristiknya

Berdasarkan definisi *agent* yang telah disebutkan pada seksi sebelumnya, terdapat dua hal penting yang dapat disimpulkan. Pertama, *agent* mempunyai kemampuan untuk melakukan suatu tugas. Kedua, *agent* melakukan tugas tersebut dalam kapasitas untuk sesuatu, atau untuk orang lain. Dua hal inilah yang mendasari Caglayan dalam mendefinisikan *software agent*. Kemudian, beberapa peneliti lain menambahkan satu hal lagi, yaitu bahwa *agent* harus bisa berjalan dalam kerangka lingkungan jaringan (*network environment*) [Brenner, Zarnekow, dan Wittig].

Tidak semua karakteristik dan atribut yang ada terangkum dalam satu *agent*, pada hakekatnya daftar karakteristik dan atribut di bawah ini merupakan hasil survei dari karakteristik yang dimiliki oleh *agent-agent* yang ada pada saat ini [Wooldridge dam Jennings; Romi; Brenner, Zarnekow, dan Wittig]

1. Otonomi
2. Intelegensi, *Reasoning*, dan *Learning*
   Dalam konsep intelegensi, ada tiga komponen yang harus dimiliki: pengetahuan dasar internal, kemampuan *reasoning*, dan kemampuan *learning*.
3. Mobilitas dan *Stationary*
4. Pendelegasian
5. Reaktif
6. Proaktif dan *Goal-oriented*
7. Kemampuan Berkomunikasi dan Berkoordinasi

Multi Agent System

Dalam perkembangan aplikasi dan penelitian tentang *agent*, bagaimanapun juga dalam suatu komunitas sebuah sistem tidak dapat dihindari akan dibutuhkannya lebih dari satu *agent*, seiring dengan semakin kompleksnya tugas yang dikerjakan oleh sistem tersebut [Romi]. *Multi Agent System* (MAS) merupakan paradigma pengembangan sistem dimana dalam suatu komunitas sistem terdapat beberapa *agent* yang saling berinteraksi, bernegosiasi, dan berkoordinasi satu sama lain dalam menjalankan suatu tugas/pekerjaan.

Interaksi antar *agent* dalam MAS terbagi menjadi empat jenis, yaitu:

1. Kerjasama
2. Koordinasi
3. Persaingan bebas
4. Persaingan ketat

Ant Colony System dan Asal Usulnya

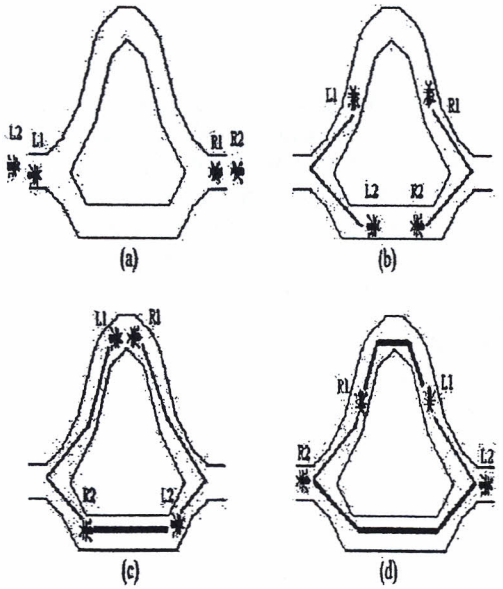
Semut dan Perilakunya

Semut adalah serangga sosial yang hidupnya berkoloni, semut secara individu tidaklah begitu berguna. Semut dapat bekerja sama dengan sesamanya secara efektif untuk melaksanakan sejumlah pekerjaan. Sebagai contoh, semut mampu untuk menemukan jalur terpendek dari suatu sumber makanan ke sarang mereka [Beckers, Deneubourg, dan Goss, 1992; Goss et al] tanpa menggunakan

petunjuk yang nyata [Hölldobler dan Wilson], dan kembali lagi ke sumber makanan tersebut. Mereka juga mampu untuk beradaptasi dengan perubahan yang terjadi di dalam lingkungan mereka, sebagai contoh menemukan jalur terpendek yang baru ketika yang lama sudah tidak memungkinkan lagi karena munculnya rintangan [Beckers, Deneubourg, dan Goss; Goss et.al.].

Hal ini telah diamati bahwa pada saat berjalan, semut telah menaruh sejumlah informasi, yang disebut *pheromone* (dalam jumlah tertentu), di tempat yang dilaluinya itu sehingga menandai jalur tersebut. Semut berikutnya yang melalui jalur tersebut dapat mengidentifikasi *pheromone* yang diletakkan oleh semut sebelumnya, memutuskan dengan probabilitas yang tinggi untuk mengikutinya, dan menguatkan jalur yang dipilihnya itu dengan *pheromone* miliknya. Perilaku mendasar semut ini dapat digunakan untuk menjelaskan bagaimana mereka dapat menemukan jalur terpendek yang baru dengan menghubungkan kembali jalur yang terputus akibat munculnya rintangan yang telah memotong jalur sebelumnya.

Bentuk komunikasi tidak langsung yang diperantarai oleh *pheromone* ini disebut *stigmergy*. Gambar 1. mengilustrasikan proses dari *stigmergy*. Semut menggunakan *pheromone* untuk menemukan jalur terpendek antara dua ujung yang dihubungkan dengan dua cabang: bawah (yang lebih pendek) dan atas (yang lebih panjang). Semut-semut memulai perjalanannya dari masing-masing ujung (Gambar 1a).



Gambar 1. Proses dari *Stigmergy*

Karena belum terdapat *pheromone* pada jalur yang ada maka semut memutuskan secara acak jalur yang mana yang akan dipilihnya. Sebagian semut memilih jalur yang bawah (semut L2 dan R2) dan sebagian yang lain memilih jalur yang atas (L1 dan R1). Saat berjalan, setiap semut menaruh *pheromone* pada jalur yang dilewatinya, yang diwakili oleh garis lurus yang terdapat pada jalur tersebut (Gambar 1b). Karena setiap semut berjalan dengan kecepatan yang tetap dan sama, semut-semut yang melewati jalur yang bawah, yang lebih pendek, telah mendekati ujung rute mereka sementara semut-semut yang melewati jalur yang atas, yang lebih panjang, baru mencapai setengah perjalanan (Gambar 1c). Dari gambar ini pula, kita dapat melihat bahwa garis yang terdapat pada jalur yang bawah lebih tebal daripada garis yang lain. Hal ini menunjukkan bahwa tingkat *pheromone* pada jalur tersebut lebih tinggi dibandingkan dengan jalur yang lain. Pada akhirnya, semut L2 dan R2 menjangkau lebih cepat ujung rute mereka (Gambar 1d). Oleh karena itu, semakin banyaklah *pheromone* yang ditaruh pada jalur yang bawah, dan membuat semut-semut baru lebih tertarik untuk melewatinya karena tingkat *pheromone*nya lebih tinggi.

Fitur penting dari perilaku semut ini disebut mekanisme *autocatalytic* (umpan balik positif). Penggabungan antara mekanisme *autocatalytic* dengan evaluasi solusi implisit digunakan dalam algoritma *ant* ini [Walkowiak]. Hal ini berarti bahwa semakin banyak semut yang mengikuti sebuah jalur maka semakin bertambah menariklah jalur tersebut untuk dilalui. Probabilitas dimana seekor semut memutuskan untuk mengikuti suatu jalur meningkat dengan banyaknya semut yang lebih dulu menggunakan jalur tersebut.

Laporan menyeluruh mengenai perilaku semut dan pengaruhnya pada algoritma *ant* dapat dilihat pada [Dorigo, Maniezzo, dan Colorni, 1996; Dorigo, Maniezzo, dan Colorni, 1991; Dorigo, Di Caro, dan Gambardella, 1999].

Ant System

Perilaku koloni semut yang telah dibahas pada subseksi sebelumnya itu telah menginspirasikan munculnya sebuah metodologi yang di dalamnya terdapat sekumpulan semut buatan, yang dinamai dengan *ants*, yang saling bekerja sama dalam mencari solusi terhadap suatu masalah optimisasi kombinatorial dengan cara bertukar informasi melalui *pheromone* yang diletakkan pada ruas-ruas sebuah graf. Oleh karena itu, sistem ini dinamai dengan *Ant System* (AS) dan algoritma yang diperkenalkan ini disebut algoritma *ant* [Dorigo, Maniezzo, dan Colorni]. Semut-semut buatan (untuk selanjutnya ditulis sebagai *ants*) yang digunakan dalam sistem ini mempunyai perbedaan besar dengan hewan semut yang asli, antara lain *ants* akan memiliki memori, mereka tidak sepenuhnya buta, dan mereka akan berada pada lingkungan dimana waktunya adalah diskrit.

Pada subseksi ini akan dibahas mengenai AS yang diterapkan pada *Traveling Salesman Problem* (TSP) berdasarkan eksperimen yang dilakukan oleh Marco Dorigo [Dorigo, Maniezzo, dan Colorni, 1991], yang merupakan penemu algoritma ini. TSP dapat dinyatakan sebagai permasalahan dalam mencari jarak minimal sebuah tur tertutup terhadap sejumlah n kota dimana kota-kota yang ada hanya dikunjungi sekali. TSP direpresentasikan dengan

menggunakan sebuah graf dimana graf tersebut merupakan graf yang lengkap, artinya semua simpulnya terhubung satu sama lain. Jadi, jika terdapat n buah simpul maka graf tersebut memiliki *(n!/((n-2)! 2!))* buah ruas, sesuai dengan rumus kombinasi, dan juga memiliki *(n-1)!/2* tur yang mungkin yang dapat dilakukan oleh setiap semut. Graf tersebut juga merupakan graf simetris, artinya jarak antara kota *r* ke kota *s* sama dengan jarak antara kota *s* ke kota *r* ($\delta(r,s) = \delta(s,r)$).

Secara informal, AS bekerja sebagai berikut. Setiap semut memulai turnya melalui sebuah kota yang dipilih secara acak (setiap semut memiliki kota awal yang berbeda). Secara berulang kali, satu-persatu kota yang ada dikunjungi oleh *ants* dengan tujuan untuk menghasilkan tur yang lengkap (yaitu mengunjungi masing-masing kota sekali saja). Pemilihan kota-kota yang akan dilaluinya didasarkan pada suatu fungsi probabilitas, dinamai aturan transisi status (*state transition rule*), dengan mempertimbangkan *visibility* (invers dari jarak) kota tersebut dan jumlah *pheromone* yang terdapat pada ruas yang menghubungkan kota tersebut. *Ants* lebih suka untuk bergerak menuju ke kota-kota yang dihubungkan dengan ruas yang pendek dan atau memiliki tingkat *pheromone* yang tinggi [Dorigo dan Gambardella, 1997]. Perlu diketahui bahwa setiap semut memiliki sebuah memori, dinamai *tabu list*, yang berisi semua kota yang telah dikunjunginya pada setiap tur. *Tabu list* ini mencegah *ants* untuk mengunjungi kota-kota yang sebelumnya telah dikunjungi selama tur tersebut berlangsung, yang membuat solusinya menjadi mungkin.

Setelah semua *ants* menyelesaikan tur mereka dan *tabu list* mereka menjadi penuh, sebuah aturan pembaruan *pheromone* global (*global pheromone updating rule*) dilaksanakan pada setiap semut: Penguapan *pheromone* pada semua ruas dilakukan, dan kemudian setiap semut menghitung panjang tur yang telah mereka lakukan lalu menaruh sejumlah *pheromone* pada ruas-ruas yang merupakan bagian dari tur mereka yang sebanding dengan kualitas dari solusi yang mereka hasilkan: Semakin pendek sebuah tur yang dihasilkan oleh seekor semut, jumlah *pheromone* yang diletakkan pada ruas-ruas yang dilaluinya pun semakin besar. Dengan kata lain, ruas-ruas yang merupakan bagian dari tur-tur yang pendek adalah ruas-ruas yang menerima jumlah *pheromone* yang lebih besar. Hal ini menyebabkan ruas-ruas yang diberi *pheromone* lebih banyak akan lebih diminati/dipertimbangkan pada tur-tur selanjutnya, dan sebaliknya ruas-ruas yang tidak diberi *pheromone* menjadi kurang diminati. Dan juga, jalur terpendek yang ditemukan oleh *ants* disimpan dan semua *tabu list* yang ada dikosongkan kembali.

Peranan utama dari penguapan *pheromone* tadi adalah untuk mencegah stagnasi, yaitu situasi dimana semua *ants* berakhir dengan melakukan tur yang sama. Proses di atas kemudian diulangi sampai tur-tur yang dilakukan mencapai jumlah maksimum (berdasarkan *user*) atau sistem ini menghasilkan perilaku stagnasi dimana sistem ini berhenti untuk mencari solusi alternatif.

Aturan transisi status yang digunakan oleh AS dinamai *random-proportional rule*, yang ditunjukkan oleh persamaan (1), merupakan probabilitas dari semut *k* pada kota *r* yang memilih untuk menuju ke kota *s*.

$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)]\cdot[\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^{\beta}} & \text{jika } s \in J_k(r) \\ 0 & \text{sebaliknya} \end{cases} \quad (1)$$

dimana τ adalah *pheromone*, $\eta = 1/\delta$ adalah *visibility* (invers dari jarak $\delta(r,s)$) dimana $(\delta(r,s) = [(x_r - x_s)^2 + (y_r - y_s)^2]^{1/2})$, $J_k(r)$ adalah kumpulan kota yang akan dikunjungi oleh semut *k* yang sedang berada pada kota *r* (untuk membuat solusinya menjadi mungkin), dan β adalah sebuah parameter yang mengontrol bobot (*weight*) relatif dari *pheromone* terhadap jarak ($\beta > 0$).

Pada persamaan (1) kita mengalikan *pheromone* pada ruas (r,s) dengan nilai *visibility* yang sesuai, $\eta(r,s)$. Dengan cara ini kita lebih memilih ruas yang lebih pendek dan memiliki jumlah *pheromone* yang lebih besar.

Dalam *ant system*, aturan pembaruan *pheromone* global diimplementasikan sebagai berikut: Setelah semua *ants* membuat tur mereka, *pheromone* yang ada pada semua ruas diperbarui menurut persamaan

$$\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s)$$

$$\text{dimana } \Delta\tau_k(r,s) = \begin{cases} 1/L_k & \text{jika } (r,s) \in \text{tur yang dilakukan oleh semut k} \\ 0 & \text{sebaliknya} \end{cases} \quad (2)$$

dimana $0<\alpha<1$ adalah sebuah parameter *pheromone decay*, L_k adalah panjang tur yang dilakukan oleh semut *k*, dan *m* adalah jumlah dari *ants*.

Mengutip dari hasil percobaan yang dilakukan oleh Dorigo, Maniezzo, dan Colorni [Dorigo, Maniezzo, dan Colorni, 1996], kekuatan AS dapat disimpulkan sebagai berikut:

- Di dalam cakupan parameter keoptimalan, algoritma ini selalu menemukan solusi yang sangat bagus untuk semua permasalahan yang dicoba.
- Algoritma ini dengan cepat menemukan solusi yang bagus; meskipun demikian ia tidak memperlihatkan perilaku stagnasi, maksudnya *ants* terus mencari kemungkinan adanya tur baru yang lebih baik.

- Dengan meningkatkan dimensi masalah, kepekaan nilai-nilai parameter pada dimensi masalah tersebut diketahui menjadi sangat rendah.

Dorigo bersama dengan Gambardella [Dorigo dan Gambardella, 1997] juga menyimpulkan sebagai berikut: Meskipun AS bermanfaat dalam menemukan solusi-solusi yang optimal ataupun bagus untuk TSP dengan jumlah kota yang sedikit (sampai dengan 30 kota), waktu yang dibutuhkan untuk mendapatkan hasil tersebut membuatnya menjadi tidak mungkin lagi untuk diterapkan pada masalah yang lebih besar. Dorigo dan Gambardella menemukan tiga perubahan besar untuk meningkatkan performa AS yang mengarah pada pendefinisian *Ant Colony System* (ACS), yang akan dibahas pada subseksi berikut ini.

Ant Colony System

Seperti yang telah diberitahukan sebelumnya bahwa ACS dibuat berdasar pada *ant system* terdahulu dengan maksud untuk meningkatkan efisiensinya ketika diterapkan pada TSP yang lebih kompleks. Akan tetapi, ACS memiliki perbedaan dengan *ant system* yang sebelumnya karena tiga aspek utama [Dorigo dan Gambardella, 1997]: (i) aturan transisi status pada sistem ini memberikan suatu cara langsung untuk menyeimbangkan antara penjelajahan (*exploration*) ruas-ruas yang baru dengan eksploitasi (*exploitation*) dari sebuah *priori* dan pengetahuan yang dihimpun mengenai masalah tersebut, (ii) aturan pembaruan *pheromone* global hanya dilakukan pada ruas-ruas yang merupakan bagian dari tur terbaik, dan (iii) disaat *ants* membangun sebuah solusi, diterapkan suatu aturan pembaruan *pheromone* lokal (*local pheromone updating rule*).

Secara informal, ACS bekerja sebagai berikut: pertama kali, sejumlah m *ants* ditempatkan pada sejumlah n kota berdasarkan beberapa aturan inisialisasi (misalnya, secara acak). Setiap semut membuat sebuah tur (yaitu, sebuah solusi TSP yang mungkin) dengan menerapkan sebuah aturan transisi status secara berulang kali. Selagi membangun turnya, seekor semut juga memodifikasi jumlah *pheromone* pada ruas-ruas yang dikunjunginya dengan menerapkan aturan pembaruan *pheromone* lokal yang telah disebutkan tadi. Setelah semua *ants* mengakhiri tur mereka, jumlah *pheromone* yang ada pada ruas-ruas dimodifikasi kembali (dengan menerapkan aturan pembaruan *pheromone* global). Seperti yang terjadi pada *ant system*, dalam membuat tur, *ants* 'dipandu' oleh informasi heuristik (mereka lebih memilih ruas-ruas yang pendek) dan oleh informasi *pheromone*: Sebuah ruas dengan jumlah *pheromone* yang tinggi merupakan pilihan yang sangat diinginkan. Kedua aturan pembaruan *pheromone* itu dirancang agar *ants* cenderung untuk memberi lebih banyak *pheromone* pada ruas-ruas yang harus mereka lewati.

Aturan Transisi Status pada ACS

Aturan transisi status yang berlaku pada ACS adalah sebagai berikut: seekor semut yang ditempatkan pada kota r memilih untuk menuju ke kota s dengan menerapkan aturan yang ditunjukkan oleh persamaan (3)

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ [\tau(r,u)] \cdot [\eta(r,u)]^{\beta} \right\} & \text{jika } q \le q_0 \\ & \text{(exploitation)} \\ S & \text{sebaliknya} \\ & \text{(biased exploration)} \end{cases} \quad (3)$$

dimana q adalah sebuah bilangan pecahan acak antara 0 s.d. 1 [0 .. 1], q_0 adalah sebuah parameter ($0 \le q_0 \le 1$ dan S adalah sebuah variabel acak yang dipilih berdasarkan distribusi probabilitas yang telah diberikan pada persamaan (1) di atas.

Aturan transisi status yang dihasilkan dari persamaan (3) dan (1) dinamakan aturan *random-proportional* semu (*pseudo-random-proportional rule*). Aturan transisi status ini, sebagaimana dengan aturan *random-proportional* terdahulu, mengarahkan *ants* untuk bertransisi ke kota-kota yang dihubungkan dengan ruas-ruas yang pendek dan memiliki jumlah *pheromone* yang besar. Setiap kali seekor semut yang ada pada kota r harus memilih kota s sebagai tujuan berikutnya, ia membangkitkan sebuah bilangan acak antara 0 s.d. 1 ($0 \le q_0 \le 1$). Jika $q \le q_0$ maka semut tersebut akan memanfaatkan pengetahuan yang ada mengenai masalah tersebut, yaitu pengetahuan heuristik tentang jarak antara kota tersebut dengan kota-kota lainnya dan juga pengetahuan yang telah didapat dan disimpan dalam bentuk *pheromone trail*. Hal ini mengakibatkan ruas terbaik (berdasarkan persamaan (3)) dipilih (*exploitation*). Jika sebaliknya maka sebuah ruas dipilih berdasarkan persamaan (1) (*biased exploration*).

Aturan Pembaruan Pheromone Global pada ACS

Pada sistem ini, pembaruan *pheromone* secara global hanya dilakukan oleh semut yang membuat tur terpendek sejak permulaan percobaan. Pada akhir sebuah iterasi, setelah semua *ants* menyelesaikan tur mereka, sejumlah *pheromone* ditaruh pada ruas-ruas yang dilewati oleh seekor semut yang telah menemukan tur terbaik (ruas-ruas yang lain tidak diubah). Tingkat *pheromone* itu diperbarui dengan menerapkan aturan pembaruan *pheromone* global yang ditunjukkan oleh persamaan (4)

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s)$$

$$\text{dimana } \Delta\tau(r,s) = \begin{cases} \left(L_{gb}\right)^{-1} & \text{jika } (r,s) \in \text{global-best-tour} \\ 0 & \text{sebaliknya} \end{cases} \quad (4)$$

$0<\alpha<1$ adalah parameter *pheromone decay*, dan L_{gb} adalah panjang dari tur terbaik secara global sejak permulaan percobaan. Seperti yang terjadi pada *ant system*, pembaruan *pheromone* global dimaksudkan untuk memberikan *pheromone* yang lebih banyak pada tur-tur yang lebih pendek. Persamaan (4) menjelaskan bahwa hanya ruas-ruas yang merupakan bagian dari tur terbaik secara global yang akan menerima penambahan *pheromone*.

Aturan Pembaruan Pheromone Lokal pada ACS

Selagi melakukan tur untuk mencari solusi dari TSP, *ants* mengunjungi ruas-ruas dan mengubah tingkat *pheromone* pada ruas-ruas tersebut dengan menerapkan aturan pembaruan *pheromone* lokal yang ditunjukkan oleh persamaan (5)

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s)+\rho\cdot\Delta\tau(r,s) \qquad (5)$$

dimana $0<\rho<1$ adalah sebuah parameter.

Peranan dari aturan pembaruan *pheromone* lokal ini adalah untuk mengacak arah tur-tur yang sedang dibangun, sehingga kota-kota yang telah dilewati sebelumnya oleh tur seekor semut mungkin akan dilewati kemudian oleh tur *ants* yang lain. Dengan kata lain, pengaruh dari pembaruan lokal ini adalah untuk membuat tingkat ketertarikan ruas-ruas yang ada berubah secara dinamis: setiap kali seekor semut menggunakan sebuah ruas maka ruas ini dengan segera akan berkurang tingkat ketertarikannya (karena ruas tersebut kehilangan sejumlah *pheromone*-nya), secara tidak langsung *ants* yang lain akan memilih ruas-ruas lain yang belum dikunjungi. Konsekuensinya, *ants* tidak akan memiliki kecenderungan untuk berkumpul pada jalur yang sama. Fakta ini, yang telah diamati dengan melakukan percobaan [Dorigo dan Gambardella, 1997], merupakan sifat yang diharapkan bahwa jika *ants* membuat tur-tur yang berbeda maka akan terdapat kemungkinan yang lebih tinggi dimana salah satu dari mereka akan menemukan solusi yang lebih baik daripada jika mereka semua berkumpul dalam tur yang sama. Dengan cara ini, *ants* akan membuat penggunaan informasi *pheromone* menjadi lebih baik: tanpa pembaruan lokal, semua *ants* akan mencari pada lingkungan yang sempit dari tur terbaik yang telah ditemukan sebelumnya.

Algoritma ACS

Berikut adalah algoritma ACS

1. /* *Initialization phase* */
For each pair (r,s) □(r,s):= □$_0$
End-for
For k:=1 to m do
Let r_{k1} be the starting city for ant k
$J_k(r_{k1})$:= {1, ..., n} - r_{k1}
/* $J_k(r_{ij})$ *is the set of yet to be visited cities for ant k in city* r_{ij} */
r_k:= r_{k1}
/* r_k *is the city where ant k is located* */
End-for

2. /* *This is the phase in which ants build their tours. The tour of ant k is stored in* $Tour_k$. */

For i:=1 to n do
If i<n then
For k:=1 to m do
Choose the next city s_k according to Eq.(3) and Eq.(1)
$J_k(s_k)$:= $J_k(r_k)$ - s_k
$Tour_k$ (i):=(r_k ,s_k)
End-for

Else
For k:=1 to m do
/* *In this cycle all the ants go back to the initial city* r_{k1} */
s_k := r_{k1}
$Tour_k$ (i):=(r_k ,s_k)
End-for
End-if

/* *In this phase local updating occurs and pheromone is updated using Eq.(5)* */
For k:=1 to m do
□(r_k ,s_k):=(1-□)□(r_k ,s_k)+ □□$_0$
r_k := s_k /* *New city for ant k* */
End-for
End-for

3. /* *In this phase global updating occurs and pheromone is updated* */
For k:=1 to m do
Compute L_k
/* L_k *is the length of the tour done by ant k* */
End-for

Compute L_{best}
/* *Update edges belonging to* L_{best} *using Eq. (4)* */
For each edge (r,s)
□(r_k ,s_k):=(1-□□(r_k ,s_k)+ □L_{best})$^{-1}$
End-for

4. If (End_condition = True) then Print shortest of Lk
Else
goto Phase 2

HASIL DAN DISKUSI

Analisis Algoritma

Seperti telah dijelaskan sebelumnya bahwa ACS berdasar pada *Ant System* terdahulu dengan melakukan perubahan-perubahan untuk meningkatkan efisiensinya jika diterapkan ke dalam masalah yang lebih besar dan rumit. Hal ini juga

mengartikan bahwa algoritma *ant* yang terdahulu berbeda dengan algoritma *ant* yang terdapat pada ACS.

Pada subseksi ini, penulis akan menganalisa algoritma *ant* yang terdapat pada ACS (lihat seksi 2) dalam usahanya untuk mencari jalur terpendek dari sebuah graf. ACS menggunakan beberapa *agent*, yang dinamai *ant*, dimana setiap *ant* melakukan turnya masing-masing mulai dari kota awal dan kembali lagi ke kota tersebut, dengan mengunjungi kota-kota yang ada hanya sekali, untuk mendapatkan hasil terbaik. Setiap *ant* memiliki properti-properti untuk menyimpan hasil turnya, antara lain: panjang tur (yang dihasilkan) dan koleksi kota-kota (yang merupakan bagian dari turnya).

Algoritma ini dimulai dengan menempatkan setiap *ant* pada kota awalnya masing-masing, yang diwakili oleh simpul yang ada pada graf tersebut. Tur yang dilakukan oleh setiap *ant* ini dimulai dari sebuah simpul awal dan melewati ruas-ruas yang menghubungkan *n* simpul yang ada kemudian kembali lagi ke simpul awal tersebut. Untuk mendapatkan hasil yang lebih beragam, sebaiknya setiap *ant* memiliki simpul awal yang berlainan, dan simpul awal untuk setiap *ant* tidak akan berubah selama algoritma ini berjalan.

Setelah ditempatkan pada simpul awalnya masing-masing, setiap *ant* memulai turnya dengan memilih simpul berikutnya yang akan dikunjungi berdasarkan persamaan (3) dan (1) pada seksi sebelumnya. Pemilihan simpul berikutnya ini dipengaruhi oleh panjangnya ruas yang menghubungkan antara simpul dimana *ant* berada saat ini dengan simpul yang akan ditujunya, dan jumlah *pheromone* yang ada pada ruas tersebut. *Pheromone*, seperti telah dijelaskan pada seksi sebelumnya, merupakan informasi yang ditinggalkan oleh *ant* yang telah lebih dahulu melewati ruas tersebut. Ruas yang lebih pendek dengan jumlah *pheromone* yang lebih besar akan mendapat prioritas yang lebih tinggi. Setelah menentukan simpul berikutnya yang akan dituju, *ant* 'berjalan' melewati ruas yang menghubungkan kedua simpul tadi, dan setelah sampai, *ant* memperbarui jumlah *pheromone* yang terdapat pada ruas yang dilewatinya itu berdasarkan persamaan (5) pada seksi sebelumnya. Kemudian *ant* memasukkan ruas dan simpul yang dilewatinya itu ke dalam properti turnya untuk menandakan bahwa ruas dan simpul tersebut merupakan bagian dari tur mereka. Pada saat ini juga, posisi *ant* telah berubah dari simpul awal menjadi simpul yang telah ia tuju dan *ant* kembali memilih simpul berikutnya yang akan dikunjungi.

Pada saat semua *ants* telah mengunjungi *n-1* simpul, simpul berikutnya yang akan dituju adalah simpul awal dari masing-masing *ant*. Setiap *ant* akan memilih ruas yang menghubungkan antara simpul yang merupakan posisinya saat ini dengan simpul awal dari masing-masing *ant* lalu memperbarui jumlah *pheromone* pada ruas tersebut dan memasukkan ruas dan simpul awal tersebut ke dalam properti turnya.

Setelah semua *ants* menyelesaikan tur mereka, panjang tur dari setiap *ant* dihitung dan dipilih yang paling pendek. Tur terpendek yang dihasilkan ini dijadikan sebagai tur terbaik pada saat ini lalu dibandingkan dengan tur terbaik yang telah dihasilkan sebelumnya. Jika panjang tur terbaik saat ini lebih pendek daripada panjang tur terbaik yang sebelumnya maka panjang tur dari *ant* yang menghasilkan tur terbaik saat ini dijadikan sebagai tur terbaik yang baru dan properti tur *ant* tersebut dimasukkan ke dalam properti dari tur terbaik, kemudian dilakukan pembaruan jumlah *pheromone* pada ruas-ruas yang merupakan bagian dari tur terbaik tersebut berdasarkan persamaan (4) pada seksi sebelumnya.

Kemudian dilakukan tes terhadap kondisi *end* yang menandakan penghentian proses pencarian jalur terpendek. Jika benar maka tur terbaik yang telah dihasilkan akan dicetak dan algoritma dihentikan. Jika sebaliknya maka proses pencarian akan dilanjutkan dan properti tur dari masing-masing *ant* dikosongkan kembali.

Dari penganalisaan terhadap algoritma *ant* ini, ada beberapa hal penting yang perlu dicatat, yaitu:

1. Untuk melakukan perbandingan antara tur terbaik saat ini dengan tur terbaik sebelumnya, pada kali pertama diperlukan sebuah nilai tur terbaik awal. Nilai tur terbaik awal ini dapat diperoleh dengan menginisialisasikannya dengan nilai maksimum dari sebuah tipe data. Hal ini bertujuan agar pada proses pencarian tur terbaik yang pertama akan tercipta nilai tur terbaik yang baru.
2. Pada proses pemilihan simpul berikutnya, yang didasarkan pada persamaan (3), diperlukan sebuah nilai parameter q_0 yang merupakan sebuah bilangan pecahan dimana $0 \leq q_0 \tilde{\leq} 1$
3. Setiap *ant* harus memiliki properti tur untuk menyimpan hasil turnya masing-masing. Properti tur ini berupa panjang tur, dan koleksi ruas-ruas dan simpul-simpul yang merupakan bagian dari tur mereka. Nilai dari masing-masing properti ini akan dikosongkan kembali setiap kali *ant* akan memulai turnya.
4. Proses pembaruan *pheromone* yang didasarkan pada persamaan (5) (aturan pembaruan *pheromone* lokal) dipengaruhi oleh dua parameter, yaitu ρ dan $\Delta\tau$. Nilai ρ sama dengan nilai parameter α (*pheromone decay*) pada persamaan (4), sedangkan nilai $\Delta\tau$ didapatkan dari invers hasil perkalian antara panjang tur yang diperoleh melalui penelusuran simpul-simpul terdekat dengan jumlah simpul yang ada pada graf tersebut. Tur yang dilakukan dengan menelusuri simpul-simpul terdekat ini mengikuti aturan yang diterapkan oleh Dorigo dan Gambardella [Dorigo dan Gambardella, 1997] dimana aturan ini telah terbukti dapat menunjukkan hasil yang bagus.

5. Proses pembaruan *pheromone* yang didasarkan pada persamaan (4) (aturan pembaruan *pheromone* global) dipengaruhi oleh dua parameter, yaitu α dan $\Delta\tau$. α adalah parameter *pheromone decay* dimana $0<\alpha<1$ dan nilai $\Delta\tau$ didapatkan dari invers terhadap panjang tur terbaik yang paling akhir yang ditemukan oleh *ants* sejak dimulainya pencarian.

Implementasi Algoritma

Langkah pertama merancang bentuk tampilan program kemudian dilanjutkan dengan mengimplementasikannya ke dalam bahasa pemrograman. Dalam menerjemahkan algoritma *ant* ke dalam kode-kode program, penulis menggunakan bahasa pemrograman Java. Selain itu, program ini dibuat dengan bantuan *framework* RePast http://repast.sourceforge.net. RePast merupakan sebuah perangkat lunak yang digunakan untuk merancang simulasi berbasis *agent* dengan menggunakan bahasa pemrograman Java. Di dalam RePast terdapat pustaka kelas-kelas (*class*) untuk merancang, menjalankan, menampilkan, dan mengumpulkan data dari sebuah simulasi berbasis *agent*.

Gambaran Umum Program

Program yang dibuat ini terdiri dari empat kelas, yaitu ACONode, ACOEdge, ACOAgent, dan ACOModel. Kelas ACONode berisi *method-method* yang digunakan dalam pembuatan simpul dan juga pencarian simpul-simpul dan ruas-ruas yang merupakan tetangga dari suatu simpul. Kelas ACOEdge, selain digunakan untuk membuat ruas-ruas, juga memiliki *method-method* yang digunakan untuk melakukan proses pembaruan *pheromone*. Setiap ruas yang terbentuk akan memiliki properti-properti yang terdiri dari panjang ruas (δ), *visibility* (η), dan bobot *pheromone* yang ada pada ruas yang bersangkutan. Kelas ACOAgent digunakan untuk membuat *ants* dan setiap *ant* akan melakukan turnya masing-masing. Dalam pelaksanaannya, kelas ini memiliki properti-properti untuk menyimpan hasil tur yang sedang berjalan, yaitu panjang tur yang telah dicapai dan simpul-simpul serta ruas-ruas yang telah dikunjungi sehingga setiap *ant* akan memiliki tur yang sah. Kelas ACOModel merupakan kelas utama dari program ini. Kelas ini menangani pertukaran informasi antar kelas, membangun kelas-kelas yang lain, dan menginstruksikan *ants* untuk membuat turnya masing-masing. Di dalam kelas inilah terdapat inisialisasi untuk semua parameter yang ada pada ACS.

Setelah dikompilasi, program dijalankan dengan mengeksekusi kelas ACOModel yang di dalamnya terdapat *method* public static void main(String[] args), yang merupakan bagian awal program Java yang dijalankan oleh Java Virtual Machine (JVM).

```
public static void main(String[] args) {
  SimInit init = new SimInit();
  SimpleModel model = new ACOModel();
  init.loadModel(model, args.length>0 ?
  args[0] : null, false); }
```

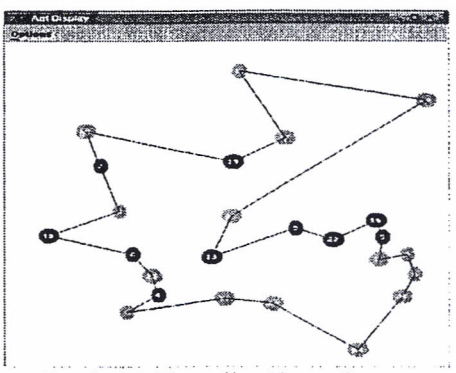
Method di atas akan menginstansiasi kelas SimInit, yang merupakan kelas bawaan dari RePast, dan juga kelas ACOModel sendiri. Kelas ACOModel, yang merupakan instans dari kelas SimpleModel, akan dijadikan sebagai model simulasi dari RePast.

Untuk memulai simulasi (pencarian jalur terpendek), dilakukan dengan menekan tombol *start* (▸) yang akan menjalankan *method* setup(), buildModel(), preStep(), step(), dan postStep() pada kelas ACOModel. *Method* setup() dan buildModel() hanya dijalankan sekali pada awal pelaksanaan simulasi dan selanjutnya kelas ACOModel hanya akan menjalankan *method* preStep(), step(), dan postStep() hingga simulasi ini dihentikan. *Method* setup() digunakan untuk inisialisasi beberapa variabel dan juga akan menjalankan *method* setup() dari kelas induk, yaitu kelas SimpleModel. *Method* buildModel() digunakan untuk membuat simpul-simpul, ruas-ruas, dan *ants* sesuai dengan nilai parameter yang ada pada form input. Simpul-simpul dan *ants* akan diletakkan secara acak dan akan ditampilkan pada form output.

Method preStep() merupakan *method* untuk mencari panjang tur yang melalui simpul-simpul terdekat dimana properti ini diperlukan pada saat dilakukannya proses pembaruan *pheromone* lokal. *Method* step() merupakan *method* yang menginstruksikan *ants* untuk melakukan tur, dan *method* postStep() merupakan *method* yang digunakan untuk mencari tur terbaik pada saat ini dan memperbarui tampilan form output apabila ditemukan tur terbaik yang baru. Dalam *method* ini juga dilakukan proses pembaruan *pheromone* global.

Untuk pembahasan proses-proses utama pada ACS yang terdapat di dalam program selengkapnya dapat dilihat pada [R.D. Raharjo, 2003].

Tampilan form output dari program ditunjukkan oleh gambar 1. berikut ini.



Gambar 2. Tampilan Form Output

Form di atas akan selalu diperbarui dengan menampilkan tur terbaik yang paling akhir yang telah dicapai oleh *ants*, sedangkan setiap kali *ants* menemukan tur terbaik yang baru, pada konsol, akan

ditampilkan *ant* yang menemukan tur terbaik tersebut beserta panjang tur dan simpul-simpul yang dilaluinya.

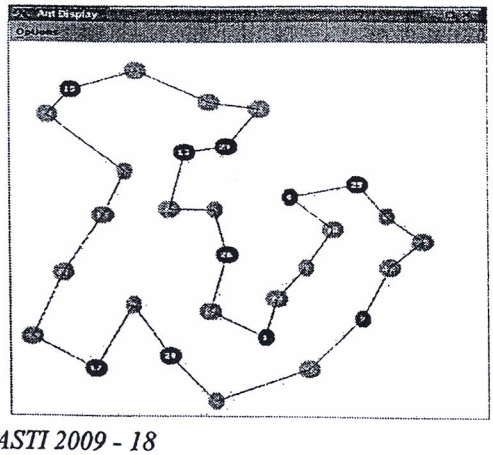
Perbandingan ACS dengan Algoritma Lain

Untuk membuktikan keoptimalannya, penulis menguji ACS dengan melakukan perbandingan antara ACS dengan *Genetic Algorithm* (GA) dan *Ant System* (AS). Dalam melakukan perbandingan ini, ACS menggunakan nilai-nilai parameter, antara lain: $\alpha=\rho=0.1$, $\beta=2$, $q_o=0.9$, dan digunakan 20 (dua puluh) *ants*. Untuk GA, digunakan populasi sebanyak 100 (seratus), sedangkan AS menggunakan parameter-parameter, antara lain: $\alpha=1$, $\beta=5$, $\rho=0.5$, Q=100, dan jumlah *ants* yang digunakan disamakan dengan yang digunakan oleh ACS, yaitu 20 *ants*. Tabel 1. menunjukkan hasil perbandingan yang telah penulis lakukan dimana untuk setiap jenis permasalahan diadakan percobaan sebanyak sepuluh kali.

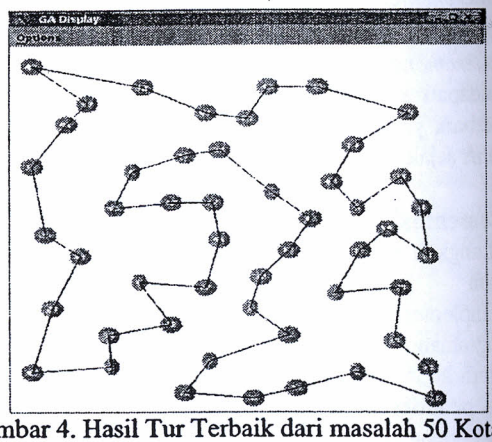
Tabel 1. Hasil Perbandingan ACS dengan Algoritma Lain

Jenis Masalah	ACS	AS	GA	Optimal
30 kota	**245** [3]	268 [225]	**245** [429]	**245**
50 kota	427 [58]	434 [904]	**426** [1894]	**426**
75 kota	**535** [55]	546 [3787]	544 [3879]	**535**
100 kota	**628** [547]	656 [3229]	639 [7844]	**628**

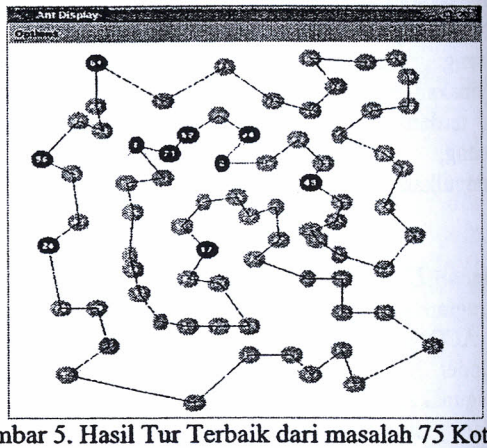
Tabel di atas memperlihatkan hasil yang diperoleh melalui percobaan yang dilakukan dimana angka yang berada diantara dua tanda kurung siku merupakan jumlah tur yang dibutuhkan untuk mendapatkan panjang tur tersebut (angka yang ada di atasnya). Tur terbaik dari setiap jenis permasalahan adalah yang dicetak tebal dan jalur yang ditempuh untuk mendapatkan tur terbaik dari setiap jenis permasalahan dapat dilihat pada gambar-gambar berikut



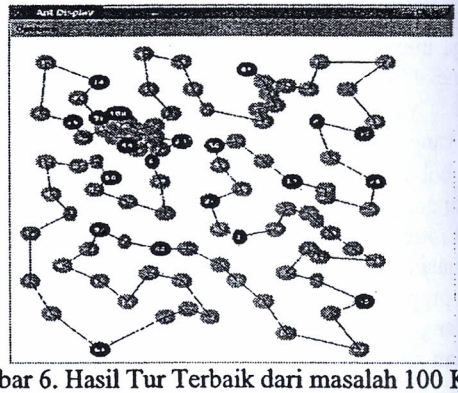
Gambar 3. Hasil Tur Terbaik dari masalah 30 Kota



Gambar 4. Hasil Tur Terbaik dari masalah 50 Kota



Gambar 5. Hasil Tur Terbaik dari masalah 75 Kota



Gambar 6. Hasil Tur Terbaik dari masalah 100 Kota

Semua jenis masalah di atas diambil dari TSPLIB (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/) yang merupakan TSP *benchmark* dengan melakukan beberapa modifikasi dalam jumlah simpul (kota)nya

KESIMPULAN

Berdasarkan percobaan yang telah dilakukan, terbukti bahwa ACS secara garis besar merupakan algoritma yang paling optimal dibandingkan dengan algoritma lain yang diuji cobakan. Dari hasil yang diperoleh dapat diketahui bahwa ACS mampu mendapatkan hasil tur terbaik dalam waktu yang relatif cepat (paling cepat jika dibandingkan dengan

dua algoritma yang lain) dan hal inilah yang membuktikan keoptimalan dari ACS. Selain itu, ACS telah membuktikan bahwa ia mampu diterapkan pada TSP yang lebih besar dan rumit dimana hal ini merupakan kelemahan dari AS, sistem yang menjadi awal mula dari ACS. Sebagai bukti, dari permasalahan 100 kota, ACS mampu menghasilkan tur terbaik yang panjangnya 628 hanya dengan melakukan tur sebanyak 547 kali.

Dengan perantara informasi *pheromone*, *ant* membentuk komunikasi secara tidak langsung dengan *ant* yang lainnya dimana akan mempengaruhi *ant* dalam memilih ruas berikutnya yang akan dilalui dan pembaruan *pheromone* yang bersifat global menyebabkan ruas-ruas yang lebih pendek akan memiliki bobot *pheromone* yang lebih besar. Hal ini menyebabkan ruas-ruas yang pendek akan lebih sering dilalui dan akhirnya akan dapat menghasilkan tur terbaik yang lebih pendek. Sedangkan, pembaruan *pheromone* yang bersifat lokal menyebabkan bobot *pheromone* di setiap ruas yang dilewati akan berubah setiap kali *ant* melewatinya dikarenakan pembaruan ini dilakukan secara *step-by-step*.

Disamping itu, teramati pula bahwa penempatan *ants* berpengaruh dalam menghasilkan tur yang lebih baik karena dari beberapa percobaan diketahui bahwa sering kali *ants* 'berhenti' menemukan tur terbaik yang baru dikarenakan semua *ants* telah menempuh jalur yang sama (perilaku stagnasi) walaupun pengaruhnya tidak terlalu besar. Tur yang dilakukan dengan pencarian terhadap ruas-ruas terdekat juga mempunyai pengaruh yang sama dimana hasil yang didapat dari penelusuran ruas-ruas terdekat ini akan mempengaruhi jumlah *pheromone* yang akan diletakkan pada saat dilakukannya aturan pembaruan *pheromone* lokal.

Kesimpulan utama dari paper ini adalah bahwa ACS menunjukkan kinerja yang lebih baik dibandingkan dua algoritma yang lain dan ACS merupakan metodologi yang patut diperhitungkan dalam mencari solusi terhadap masalah optimisasi.

DAFTAR PUSTAKA

A. Caglayan, Colin Harrison, Alper Caglayan, dan Colin G. Harrison, *Agent Sourcebook: A Complete Guide to Desktop, Internet, and Intranet Agents*, John Wiley & Sons Inc., 1997.

B. Hölldobler dan E. O. Wilson, *The Ants*, Springer-Verlag, Berlin, 1990.
Shortest Path by the Ant Lasius Niger, Journal of theoretical biology, 1992, hal. 397-415.

S. Goss, S. Aron, J. L. Deneubourg, dan J. M. Pasteels, *Self-organized Shortcuts in the Argentine Ant*, Naturwissenschaften, 1989, hal. 579-581.

Isak Rickyanto, *Dasar Pemrograman Berorientasi Objek Dengan Java 2 (JDK 1.4)*, Edisi Pertama, ANDI, Yogyakarta, 2003.

David B. Guralnik, *Webster's New World Dictionary*, Prentice Hall School Group, 1983.

Krzysztof Walkowiak, *Graph Coloring Using Ant Algorithms*, 2001.

M. Dorigo, G. Di Caro, dan L. M. Gambardella, *Ant Algorithms for Discrete Optimization*, Artificial Life, No. 5(2), 1999, hal. 137-172.

M. Dorigo, V. Maniezzo, dan A. Colorni, *Distributed Optimization by Ant Colonies*, Proceedings of ECAL91 - European Conference on Artificial Life, Paris, France, 1991, F. Varela and P. Bourgine (Eds.), Elsevier Publishing, hal. 134–142.

M. Dorigo, V. Maniezzo, dan A. Colorni, *Positive Feedback As A Search Strategy*, (Tech. Rep. 91-016), Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica, 1991.

M. Dorigo, V. Maniezzo, dan A. Colorni, *The Ant System: Optimization By A Colony of Cooperating Agents*, IEEE Transactions on Systems, 1996.

M. Dorigo dan L. M. Gambardella, *Ant Colonies for the Traveling Salesman Problem*, 1997.

M. Dorigo dan L. M. Gambardella, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, 1997.

Michael J. Wooldridge dan Nicholas R. Jennings, *Intelligent agents: Theory and Practice*, Knowledge Engineering Review, 10(2), 1995.

Romi S. Wahono, *Pengantar Multi Agent System (MAS)*, http://www.ilmukomputer. com, 12-08-2003.

R.D. Raharjo, Skripsi, Teknik Informatika Fakultas Teknologi Industri, Universitas Gunadarma, 2003

R. Beckers, J. L. Deneubourg, dan S. Goss, *Trails and U-turns in the Selection of the*

Walter Brenner, Rudiger Zarnekow, dan Hartmut Wittig, *Intelligent Software Agents: Foundation and Applications*, Springer-Verlag, 1998.

STIKOM
SURABAYA

Diselenggarakan oleh :
Bagian Penelitian Akademik (PA) STIKOM
Jl. Raya Kedung Baruk 98 Surabaya
Telp. 031. 8721731, Fax. 031.8710218
url : http://snasti.stikom.edu
email : snasti@stikom.edu

ISBN 978-979-8968-30-3
9 789798 968303